Prises de vues automatiques

J.-M Friedt `friedtj@free.fr`, É. Carry `milou@achelem.org`
Association Projet Aurore, Besançon

> "... it is well known that a vital ingredient of succes is not knowing that what you're attempting can't be done."
> Terry Pratchett, Discoworld vol. 3 (Equal Rites)

Nous proposons diverses approches permettant des prises de vues à intervalles de temps prédéfinis afin de suivre l'évolution à long terme d'évènements lents dont la dynamique est difficile à appréhender au quotidien. Nous présenterons diverses solutions fournissant des puissances de calcul décroissantes et donc des fonctionnalités de plus en plus simples mais surtout diminuant à chaque étape la consommation électrique et donc permettant une autonomie plus grande. Nous irons ainsi de la carte d'acquisition sur PC à l'appareil photographique numérique (APN) contrôlé par USB et RS232 pour finalement atteindre une solution de très faible consommation où l'APN est totalement contrôlé par un microcontrôleur réveillé par une horloge temps réel dédiée à ce type d'applications.

1 Introduction

Dès l'apparition des premières webcams, avec la disponibilité d'outils de capture d'images en ligne de commande, la possibilité de réaliser des prises de vues à intervalles de temps réguliers afin de faire des films accélérés est apparue évidente [1].

Les webcams ont cependant été dans un premier temps limitées à de la prise de vue statique en tons de gris et surtout à une résolution médiocre. Nous nous sommes alors tournés vers l'utilisation de cartes d'acquisition vidéo associées à une caméra [1]. Une originalité de notre approche a été l'utilisation d'un relais pour sélectionner deux vues d'un même évènement. Cette solution est d'autant plus attractive que le prix des cartes d'acquisition vidéo ne cesse de chuter, que des outils en ligne de commande sont disponibles pour une insertion dans un script, et finalement que la prise de vue peut être contrôlée par un événement au lieu d'une prise de vue périodique.

Ces solutions ont cependant l'inconvénient majeur d'immobiliser un ordinateur pendant la durée de l'acquisition, avec la consommation électrique associée. Nous nous sommes donc plus récemment tournés vers les appareils photo numériques (APN) qui fournissent une bien meilleure résolution pour un coût actuellement inférieur à 100 euros. Nous présenterons deux approches à l'utilisation d'APNs : la première utilisant le contrôle par ports USB ou RS232 ne nécessitant pas de modification de l'appareil, et la seconde nécessitant de démonter l'appareil photo afin de simuler l'appui des différentes touches qu'utiliserait un opérateur humain pour effectuer la prise de vue.

2 La carte d'acquisition vidéo sur PC

La solution la plus triviale à la prise de vue consiste à connecter un périphérique capable d'acquérir une image – webcam ou carte d'acquisition vidéo – et de lancer périodiquement une commande de prise de vue. Une application plus intéressante de cette approche justifiant la consommation électrique énorme d'un PC et l'immobilisation d'un ordinateur pendant la durée de l'expérience est le déclenchement de la prise de vue sur un événement. Le programme `motion` disponible à `http://www.lavrsen.dk/twiki/bin/view/Motion/WebHome` permet de déclencher la prise de vue en fonction d'un événement, qu'il s'agisse de l'écoulement d'un intervalle de temps pré-défini ou lorsqu'un mouvement est détecté dans le champ de vision de la caméra. Il se connecte à tout périphérique vidéo suivant la norme `video4linux`.

`motion` est capable de gérer indépendemment plusieurs périphériques vidéos au moyen de fichier de configuration distincts. Nous nous placerons dans le cas d'une webcam usb en mode détection de mouvement et d'une caméra analogique en prise de vue à intervalles réguliers.

Pour commencer, nous allons détailler le fichier de configuration général `/etc/motion.conf` présenté dans le tableau 1.

```
daemon off
quiet on

thread /etc/motion/cam0.conf
thread /etc/motion/cam1.conf

# Image size in pixels
width 640
height 480

framerate 25
quality 85
auto_brightness off
```

TAB. 1 – Le fichier `/etc/motion/motion.conf` permettant de lancer des captures depuis plusieurs sources différentes.

`daemon off` nous permet de ne pas lancer `motion` en mode démon et ainsi pouvoir affiner les réglages de la prise de vue. `quiet on` évite d'avoir un "beep" à chaque détection de mouvement, suivant le sujet observé cela peut être ennuyeux (souris, cambrioleur...). Nous définissons ensuite les deux fichiers de configurations de nos caméras puis nous donnons les caractéristiques communes voulues pour nos films et images.

Dans notre exemple, nous utilisons une webcam Philips ToUCam II avec les pilotes de Luc Saillard [2] : nous passons ainsi d'une résolution 160x112 pixels à une résolution 640x480 et augmentons le débit d'images. Une fois la webcam fonctionnelle, nous utilisons un logiciel de visualisation comme `camstream` afin de positionner la caméra correctement. Il est important de noter que lors de l'utilisation d'une caméra USB 1.1, on consomme la quasi totalité de la bande passante du contrôleur USB, il est donc illusoire de vouloir brancher deux webcam autrement qu'en utilisant de l'USB2 ou en ajoutant un contrôleur additionnel. Nous rentrons maintenant dans le vif du sujet avec la configuration de la caméra décrite dans le tableau 2.

```
#device de la webcam USB
videodevice /dev/video2

#nombre de pixels chang\'es
threshold 4000

#Temps minimum en seconde de fin d'evenement
gap 10

#repertoire de stockage des donnees
target_dir /home/milou/motion/cam0/

#mode film active
ffmpeg_cap_new on

#tampon rotatif d'image avant capture
pre_capture 5

#nombre d'image ajoutees en fin de capture
post_capture 10

#codec de sortie
ffmpeg_codec mpeg4

#nom des films de sortie
ffmpeg_filename cam0-%v-%Y%m%d%H%M%S

#enregistrement des images du mouvement
output_normal on

#texte supplementaire incruste sur l'image
text_left cam0
```

TAB. 2 – Le fichier `/etc/motion/cam0.conf` appelé par `motion.conf` présenté auparavant (Tab. 1). Ce port USB est connecté à une webcam.

Il y a deux aspects à regarder dans cette configuration, le seuil de détection d'un mouvement et le stockage des données. L'option cruciale de `motion` est le `threshold`, qui définit le nombre de pixels changeant entre 2 images, indiquant s'il y a ou non un mouvement. Cette valeur est à régler avec soin et dépendra de la résolution de l'acquisition (nombre total de pixels disponibles) et du type de mouvement à détecter (détecter une souris ou un éléphant dans la même pièce). La méthode usuelle est de désactiver tous les filtres de débruitage, régler un premier niveau de

détection puis d'activer et régler les uns après les autres les différents filtres disponibles (afin par exemple d'éliminer le mouvement des feuilles d'un arbre). Il est aussi possible de n'afficher que les pixels du mouvement avec les options `ffmpeg_cap_motion` et `output_motion` ou encore de dessiner une boite autour de la zone de mouvement grâce à l'option `locate` (Fig.1)

FIG. 1 – Souris prise sur le fait : le cadre sur l'image est indicateur de la zone où `motion` a détecté le mouvement.

Maintenant que le mouvement est correctement détecté, il faut le stocker. `motion` permet de générer automatiquement des fichiers vidéos des mouvements détectés à l'aide de `ffmpeg` et de prendre des images de ces mouvement afin de pouvoir faire du post-traitement (faire de la détection de forme ou des mesures par exemple). Nous avons paramétré les deux solutions. Nous activons la génération de vidéo par l'option `ffmpeg_cap_new on` et configurons le nom des fichiers générés et leur codec par les options `ffmpeg_filename` et `ffmpeg_codec` (à noter que seuls des codecs de type `mpeg4` sont disponibles). Il est possible de donner un don de prémonition à votre caméra ! En fait, le réglage de l'option `pre_capture` va générer un tampon rotatif d'images qui sera placer au début de la vidéo pour donner un effet de pause juste avant le début du mouvement. Il faut utiliser avec parcimonie cette option, une valeur au-delà de 5 risquera de faire perdre les premières images du mouvement... À l'opposé, il est intéressant d'utiliser `post_capture` pour ajouter des images après la fin du mouvement et ainsi rendre fluide la fin de la vidéo.

FIG. 2 – Souris à table

Il peut être intéressant d'enregistrer toutes les images composant le mouvement pour pouvoir les réutiliser pour du traitement d'image (ou pour les réutiliser dans un article : Fig. 2). Pour cela nous activons l'option `output_normal` à `on`. Bien entendu, `motion` dispose de plusieurs options pour configurer le nom de ces images, leur qualité, leur orientation ou encore leur format. Nous

verrons avec l'exemple suivant comment traiter et générer simplement des vidéos à partir de ces images. `motion` incruste d'office la date et l'heure dans un coin de l'image et vous laisse toute liberté pour ajouter du texte, le supprimer ou en placer dans les quatres coins de l'écran. Nous avons ici sagement placé une indication sur le numéro de caméra dans un coin. Les films issus de cet exemple sont disponibles sur `http://projetaurore.assos.univ-fcomte.fr/media/video/`.

Nous nous intéresserons maintenant à la configuration d'une caméra analogique au travers d'une carte d'acquisition vidéo et son utilisation pour prendre des images à intervalles de temps réguliers.

Nous avons utilisé un caméscope VHS SECAM auquel nous avons supprimé la mise en veille lorsqu'il n'y a pas de cassette et nous l'avons connecté à une carte PCI d'acquisition vidéo Pinnacle PCTV à base de circuit BT878. Chaque type de carte d'acquisition met à disposition un certain nombre de types d'entrées analogiques comme par exemple l'entrée tuner, S-VHS ou composite. Il faudra donc indiquer à `motion` dans son fichier de configuration le type d'entrée utilisée, sa norme (PAL, SECAM ou NTSC), puis activer l'acquisition à intervalles de temps réguliers.

```
# carte pctv bt878
videodevice /dev/video1

# entree composite, pour le s-video choisir 2
input 1

#norme SECAM
norm 2

# repertoire de stockage
target_dir /home/milou/motion/cam1/

#intervalle de temps entre images de la video
ffmpeg_timelapse 60

#mode de prise de vue (chaque heure, jour, mois, etc...)
ffmpeg_timelapse_mode manual

#motif pour le nom de la video
timelapse_filename cam1-%v-%Y%m%d-timelapse

#intervalle de temps entre chaque image en seconde
snapshot_interval 60

#motif du nom des images
snapshot_filename cam0-%v-%Y%m%d%H%M%S-snapshot

text_left cam1
```

Tab. 3 – Le fichier `/etc/motion/cam1.conf` appelé par `motion.conf` présenté auparavant (Tab. 1). Ce port est connecté à une carte d'acquisition vidéo capturant les images depuis un caméscope.

Le type d'entrée est configuré à travers l'option `input` qui dans notre cas (signal composite) est placée à `1`. Il est à noter que le type par défaut est `8`, celui nécessaire pour une webcam USB. Notre caméscope étant de norme SECAM nous positionnons l'option `norm` à `2`.

Il nous reste maintenant à configurer la prise de vue. Nous avons deux choix possibles. Le premier est d'utiliser les capacités de `motion` à prendre des images à intervalles réguliers et de les concaténer automatiquement dans un fichier vidéo au format `mpeg1`. Ceci est rendu possible par l'option `ffmpeg_timelapse_mode` de prendre des images toutes les heures, tous les jours, tous les lundis , etc... Et il est possible de le mettre en mode manuel. Nous pouvons alors définir un temps adapté au sujet filmé. Nous avons filmé une vue depuis notre local. Pour avoir le mouvement des nuages, nous avons choisi un intervalle de 60 secondes. Cette méthode n'est pas la plus efficace, le `mpeg1` généré est d'une qualité assez décevante. C'est pourquoi nous proposons une deuxième solution où nous enregistrons une image toutes les minutes et nous générons nous même plus tard la vidéo (attention, vous pouvez assez vite saturer votre disque dur !). Ceci nous permet par exemple de supprimer les vues prises la nuit qui ne sont pas forcément très palpitantes (sauf quand la Lune se lève dans l'axe de la caméra). Nous avons donc renseigné les options `snapshot_interval` et `snapshot_filename` de notre fichier de configuration.

Une fois l'acquisition effectuée, nous nous retrouvons avec un certain nombre de fichiers JPEG à traiter. Nous allons pour cela utiliser les deux couteaux suisses de la vidéo que sont `transcode` [3] et `mencoder`[4]. Nous allons tout d'abord générer un fichier contenant la liste ordonnée des `jpeg` à traiter (d'où l'avantage de correctement générer les noms des fichiers).

```
ls *.jpg > list
```

Puis nous allons utiliser cette liste avec transcode :

`transcode -i list -x imlist,null --use_rgb -z -H 0 -g 640x480 -y ffmpeg,null -F mpeg4 -o film.avi`

`-i list` indique le fichier contenant la liste des images. `-x imlist,null` indique les module d'importation vidéo et audio à utiliser. `--use_rgb` est utilisé car le module vidéo d'importation d'image `imlist` ne supporte pas le YUV. `-z` permet de retourner la vidéo tête en bas (il arrive que `transcode` retourne les images). `-H 0 -g 640x480` indique de ne pas tenter d'autodétection de la taille des images : `transcode` échoue pour une liste d'images, c'est pourquoi nous lui imposons la taille avec `-g`. Finalement, `-y ffmpeg,null -F mpeg4` définit le module d'exportation (ici `ffmpeg`) et le codec utilisé, et `-o film.avi` indique évidemment le nom du fichier résultant.

Il est donc possible de générer un fichier `list` qui ne corresponde i qu'à une partie des images enregistrées. Par exemple, nous avons laissé allumée la caméra 6 jours d'affilés et nous voullions enlever les images de la nuit. Imaginons par exemple que nous voudrions éliminer les images de la nuit du 5 septembre au 6 septembre en considérant la nuit de 21h à 6h du matin :

`rm cam1-0?-200609052[1-3]* && rm cam1-0?-200609060[0-5]*`

Il est donc facile de scripter la génération du fichier `list`.

Deux remarques tout de même, l'encodage d'une vidéo où s'alternent des images de nuits et de jours nous a posé quelques problèmes. En effet, au moment de la transition nuit/jour, l'image se pixellise. Ceci est sûrement dû à la méthode de compression utilisée. Une parade possible est de générer indépendemment les différentes transitions et de joindre ensuite les fichiers vidéos à l'aide de `mencoder`

`mencoder -ovc copy -oac copy -o out.avi file1.avi file2.avi`

De plus, vous pouvez avoir généré un grand nombre d'images et obtenir l'erreur `bash : /bin/ls : Liste d'arguments trop longue`. Dans ce cas [1], il faut passer par `find` et `xargs` : `find . -name 'cam*.jpg'|xargs ls>list`.

Le film généré avec cet exemple est disponible sur `http://projetaurore.assos.univ-fcomte.fr/media/video/`

Pour conclure cette partie, `motion` est capable de bien plus encore comme par exemple de faire du tracking (il peut piloter un moteur pour suivre une personne avec la caméra) ou encore de la caméra IP. Le principal inconvénient de cette première solution est d'immobiliser un PC pour la durée de l'expérience, avec le bruit et la consommation électrique associés. Nous avons par exemple constaté qu'un ordinateur basé sur un Pentium III cadencé à 800 MHz et équipé d'un disque dur de 2,1 GB et de 256 MB de RAM consomme 180 mA sous 220 V lors des acquisitions vidéo, soit une consommation constante d'environ 40 W.

3 L'appareil photo numérique avec port USB

Les appareils photographiques numériques (APN) récents, équipés de ports USB, possèdent généralement deux modes de fonctionnement, l'un les faisant apparaître comme unité de stockage de masse (accessible alors par le module `usb-storage` des noyaux 2.4 et 2.6) et l'autre permettant non seulement de rapatrier les fichiers contenant les images mais aussi l'envoi de commandes telles que l'activation du flash ou l'ordre de prise de vue (mode nommé PTP). Ce second mode nous intéresse pour la prise de vues automatique à intervalles de temps pré-définis.

Peu de systèmes informatiques présentent des ports USB maîtres (*USB host*), la majorité des microcontrôleurs possédant un port USB n'étant qu'esclave (type clavier, souris ou webcam par exemple). Cependant, l'immobilisation d'un ordinateur possédant un port USB et la consommation électrique associée ne saurait être justifiée pour une expérience de plus de quelques jours. Aussi proposons nous de contrôler un APN par son port USB depuis la carte FOX commercialisée par ACME *via* Lextronic en France (`http://www.lextronic.fr/fox/fox.htm`). Nous verrons que la consommation totale de ce système est de l'ordre du watt (250 mA sous 5 V pour l'ordinateur, excluant la consommation de l'APN) : sans prétendre à une application totalement autonome, la consommation est nettement plus raisonnable que celle d'un ordinateur personnel (Fig. 3).

[1] voir `http://www.gnu.org/software/coreutils/faq/#Argument-list-too-long` pour un diagnostic du prolème

La carte FOX fournit deux ports USB maîtres ainsi qu'une interface ethernet contrôlés par un processeur ETRAX (architecture CRIS) cadencé à 100 MHz, capable de faire fonctionner un système GNU/Linux complet depuis ses 16 MB de RAM et 4 MB de mémoire flash. Elle s'alimente par une tension régulée sous 5 V. La principale difficulté que nous allons aborder ici est la cross-compilation d'outils à destination du processeur embarqué CRIS depuis un PC équipé d'un processeur compatible Intel. L'outil que nous désirons compiler est `gphoto2` (`http://www.gphoto.org/`) qui fournit une interface en ligne de commande pour communiquer en mode PTP avec une large gamme d'APNs.

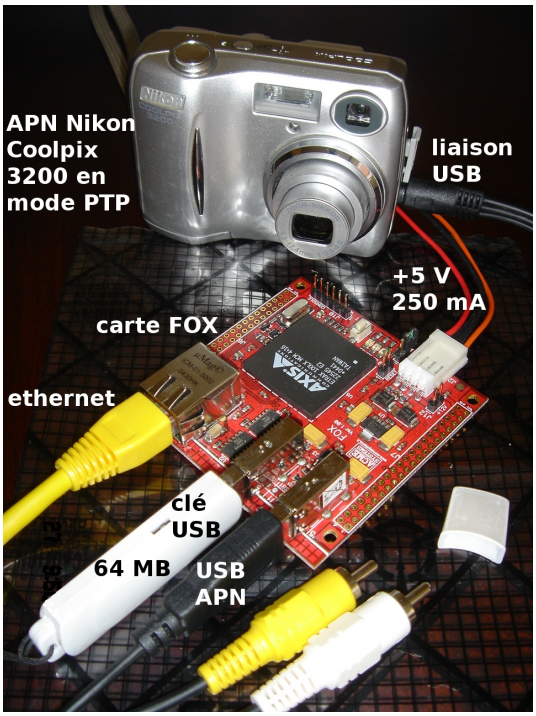

FIG. 3 – Circuit de contrôle d'un APN en mode PTP par USB. Les connecteurs RCA blanc et jaune présents sur le câble lié à l'APN ne sont pas utilisés dans cette applications.

Une rapide étude de l'arborescence issue d'une compilation de `gphoto2` sur PC montre qu'il nous sera impossible de stocker l'ensemble des programmes associés dans les 4 MB de mémoire flash de la carte FOX. Aussi déciderons nous dès maintenant de compiler le programme sur une clé USB et d'exécuter ultérieurement depuis ce support. En effet, par défaut la carte FOX est fournie avec un noyau capable de communiquer sur le port USB avec les périphériques de stockage de masse tels que les clés.

L'étude de l'arborescence de la carte FOX nous montre que les points de montage des périphériques se situent dans `/mnt` avec une sous arborescence numérotée, à côté de `/mnt/flash` qui fournit un petit espace de mémoire non-volatile embarquée. Nous monterons ultérieurement la clé USB contenant notre programme dans `/mnt/1` sur la carte FOX. Il faut donc dupliquer ce point de montage `/mnt/1` sur le PC sur lequel se fera la cross-compilation et y monter la clé USB que nous utiliserons sur la carte embarquée.

La cross-compilation de `gphoto2` et des librairies associées pour la carte FOX nécessite :

1. les outils de compilation à destination du processeur ETRAX tels que disponible à `http://www.acmesystems.it/?id=701` (fichier `cris-dist_1.63-1_i386.deb`),
2. un environnement de compilation d'une nouvelle image à flasher dans la carte FOX tel que disponible sur la même page (fichiers `devboard-R2_01-distfiles.tar.gz` et `devboard-R2_01.tar.gz`

3. les archives de gphoto2 (`gphoto2-2.2.0.tar.gz`), les librairies associées (`libgphoto2-2.2.1.tar.gz` et `libusb-0.1.7.tgz`) ainsi que les outils associés (`popt-1.7.tar.gz`). Mieux vaut éviter la distribution de `libusb` mise à disposition sur le site web de ACME qui semble manquer de fonctionnalités.

De façon générale, la cross-compilation d'un programme à destination d'un processeur d'architecture CRIS depuis un PC (processeur compatible Intel) s'obtiendra par l'incantation
`./configure --host=cris-axis-linux-gnu --without-exif --prefix=/mnt/1/gphoto --with-drivers=ptp2 && make && make install`
qui signifie que nous allons utiliser les outils de compilation `cris-axis-linux-gnu` (placés par défaut dans le répertoire `/usr/local/cris`) afin de compiler des programmes qui seront placés dans le répertoire `/mnt/1/gphoto`), emplacement auquel nous aurons pris soin auparavant de monter la clé USB qui nous servira de support (en fait la clé est montée dans `/mnt/1` et contient un répertoire `gphoto`). Les deux options `--with-drivers=ptp2` et `--without-exif` sont spécifiques à `gphoto` mais seront simplement ignorées pour les autres compilations.

Dans l'ordre :

1. exécuter le script d'initilisation des chemins `init_env` dans le répertoire contenant l'archive permettant de générer l'image GNU/Linux à destination du processeur CRIS,
2. nous compilons `popt-1.7.tar.gz`
3. nous compilons `libusb-0.1.7.tgz`
4. nous définissons les variables nécessaires pour la suite des opérations :`export LIBUSB_CFLAGS= -I/usr/local/cris/include` et `export LIBUSB_LIBS="-L/mnt/1/gphoto/lib/ -fPIC -lusb"` pour la libusb et de même pour popt (`export POPT_LIBS="-L/mnt/1/gphoto/lib/ -lpopt"` ...
5. nous compilons `libgphoto2-2.2.1.tar.gz`
6. finalement nous compilons `gphoto2-2.2.0.tar.gz`
7. en l'état `gphoto2` ne fonctionne pas sur la carte FOX car `gphoto2` recherche par défaut les librairies définissant les ports par lesquels il peut communiquer ainsi que les protocoles supportés dans le même répertoire que l'emplacement des librairies lors de la compilation (dans notre cas `/usr/local/cris/lib`). Nous allons donc recompiler une nouvelle image pour la carte FOX tel que décrit à `http ://www.acmesystems.it/?id=701` (par `./install && ./configure && make`). Cependant nous allons modifier l'image placée dans `devboard-R2_01/target/cris-axis-linux-gnu` en ajoutant un lien symbolique de `/usr/local/cris` vers `/mnt/1/gphoto` afin que `gphoto2` trouve les librairies dans le répertoire escompté. En effet l'exécution de `gphoto2` en l'état avec l'option `debug` nous montre que

```
[root@axis /mnt/1/gphoto/bin]222# ./gphoto2 --capture-image
0.006519 main(2): gphoto2 2.2.0
0.011867 main(2): gphoto2 has been compiled with the following options:
0.017128 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.022528 main(2):  + popt (for handling command-line parameters)
0.027708 main(2):  + no exif (for displaying EXIF information)
[...]
0.044238 main(2): libgphoto2 2.2.1
0.045132 main(2): libgphoto2 has been compiled with the following options:
0.045845 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.046609 main(2):  + no EXIF (for special handling of EXIF files)
0.047389 main(2):  + no /proc/meminfo (adapts cache size to memory available)
0.048337 main(2): libgphoto2_port 0.6.1
0.049219 main(2): libgphoto2_port has been compiled with the following options:
0.050023 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.050716 main(2):  + USB (libusb, for USB cameras)
0.051555 main(2):  + serial (for serial cameras)
```

```
[...]
0.069425 gphoto2-port-info-list(2): Using ltdl to load io-drivers from '/usr/loc
al/cris/lib/libgphoto2_port/0.6.1'...
0.075444 gphoto2-port-info-list(2): Called for filename '/usr/local/cris/lib/lib
gphoto2_port/0.6.1/ptpip'.
[...]
```

L'image résultante est réassemblée par `make fimage` suivi de `boot_linux -F` exécuté après avoir fermé le jumper JP8 et effectué un reset (ou une remise sous tension) pour flasher la nouvelle image en mémoire non-volatile.

8. pour des raisons que nous ne saurions expliquer, le port `disk` crée une erreur de segmentation lors de la recherche des périphériques par `gphoto2`. Nous effaçons donc toute référence à ce port : `rm -rf /mnt/1/gphoto/lib/gphoto2_port/0.6.1/disk*` (il reste alors les ports `ptpip`, `serial` pour le RS232 et `usb` qui nous intéresse ici).
9. finalement la commande `gphoto2 --list-ports` identifie les ports accessibles pour communiquer avec un APN :

```
Devices found: 7
Path                                Description
----------------------------------------------------------
ptpip:                              PTP/IP Connection
serial:/dev/ttyS0                   Serial Port 0
serial:/dev/ttyS2                   Serial Port 2
serial:/dev/ttyS3                   Serial Port 3
usb:                                Universal Serial Bus
usb:001,007                         Universal Serial Bus
usb:001,008                         Universal Serial Bus
```

La présence des ports `usb` et des deux périphériques associés (APN et clé USB) nous garantit le bon fonctionnement de l'application.

Une archive des programmes et librairies compilés et prêts à être placés sur une clé USB est disponible à `http://jmfriedt.free.fr`.

À ce point, les commandes classiques disponibles avec `gphoto2` sous PC sont accessibles, par exemple `gphoto2 --capture-image` pour demander à l'APN de prendre une image. Une telle ligne de commande peut par exemples est appelée périodiquement depuis un crontab (`http://www.acmesystems.it/?id=58`).

4 L'appareil photo numérique avec port RS232

Nous avons vu que la carte FOX consomme encore près de 250 mA sous 5 V : un accumulateur ou pile alcaline classique ne fournissant qu'une capacité de l'ordre de 3 à 10 A.h, l'autonomie de ce montage serait au mieux d'une demi journée à deux jours, sans grand intérêt pour un montage devant être capable de fonctionner en l'absence d'alimentation externe. Avant l'apparition des APN avec ports USB, une large gamme d'appareils photo avec des résolutions honorables (de l'ordre du Mpixel) ont été produits avec un port série. L'application en ligne de commande pour transmettre des informations à certain de ces appareils est `photopc`.

Nous avons ici réalisé nos expériences avec un Olympus C860L connecté par un câble série réalisé suivant la description disponible à `http://www.camerahacker.com/Olympus_C-2500L/serial_pin-out.html`. Cet appareil se démonte par ailleurs facilement pour accéder à un switch sous le capot mobile en face avant dont la fermeture active la mise en marche de l'appareil : nous verrons plus bas comment commander un tel interrupteur de façon logicielle (section 5).

L'auteur de `photopc` a non seulement fourni un mode verbeux dans lequel le logiciel informe l'utilisateur de toutes les transactions sur le port RS232, mais a aussi rédigé une excellente description du protocole de communication (`http://photopc.sourceforge.net/protocol.html`)

```
// Minimal protocol for taking pictures with the Olympus Camedia C860L
// Initial RS232 communication baud rate MUST be 19200 (init + set speed)

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h> /* declaration of bzero() */
#include <fcntl.h>
#include <termios.h>

#define BAUDRATE B19200                          // #define BAUDRATE B9600
#define RS_DEVICE "/dev/ttyS1"
#define DEBUG printf

struct termios oldtio,newtio;
extern struct termios oldtio,newtio;

int init_rs232()
{int fd;
 fd=open(RS_DEVICE, O_RDWR | O_NOCTTY );
 if (fd <0) {perror(RS_DEVICE); exit(-1); }
 tcgetattr(fd,&oldtio);               /* save current serial port settings */
 bzero(&newtio, sizeof(newtio));      /* clear struct for new port settings */
 newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;          /* _no_ CRTSCTS */
 newtio.c_iflag = IGNPAR;
 newtio.c_oflag = ONOCR;
 newtio.c_cc[VTIME] = 0;                  /* inter-character timer unused */
 newtio.c_cc[VMIN] = 1;          /* blocking read until 1 character arrives */
 tcflush(fd, TCIFLUSH);tcsetattr(fd,TCSANOW,&newtio);
 return(fd);
}

void free_rs232(int fd)
{tcsetattr(fd,TCSANOW,&oldtio);close(fd);} /* restore the old port settings */

int main(int argc,char **argv)
{int fd,k;char buf[255],
    buf1[255]={0x1B,0x53,0x06,0x00,0x00,0x11,0x02,0x00,0x00,0x00,0x13,0x00},
                                        // cmd17=setspeed, arg=2=19200 bauds
    buf2[255]={0x1b,0x43,0x03,0x00,0x02,0x02,0x00,0x04,0x00};
                                       // cmd02=snaphshot
 fd=init_rs232();
 buf[0]=0x00;write(fd,buf,1);DEBUG("cmd(%x)",(buf[0]&0xff));// init comm
 read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));            // answers 0x15
 write(fd,buf1,12);DEBUG("cmd(%x)",(buf1[5]&0xff));      // baud rate=19200
 read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));            // ans: 0x06 (ACK)
// for (k=1;k<3;k++) {
   write(fd,buf2,9);DEBUG("snapshot(%x)",(buf2[5]&0xff));// take pict
   read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));       // answers 0x06 (ACK)
   read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));       // ans: 0x05 (done)
//   printf("Waiting 5 seconds ... ");sleep(5);
   printf("done\n"); // }
 free_rs232();
 return(0);
}
```

TAB. 4 – Programme de contrôle d'un appareil photo numérique muni d'un port RS232 depuis GNU/Linux, fonctionnel sous uClinux.

que nous pouvons alors implémenter sur tout microcontrôleur faible consommation muni d'un port série. Cette solution a dans un premier temps été implémentée sous GNU/Linux telle que présenté dans le tableau 4, et sa fonctionnalité a été validée sous uClinux lors d'une utilisation sur la carte uCdimm5272 [6] à base de Coldfire 5272 commercialisée par Arcturus Networks [2], puis dans un microcontrôleur compatible 8051 (code dans le tableau 5). La seule subtilité dans cette dernière implémentation du code est le passage du cœur du processeur à une cadence élevée de 16 MHz afin de pouvoir générer les signaux de communication à 19200 bauds.

5 Simulation de l'appui de touches sur appareil photo numérique

Les appareils munis d'un port série étant malheureusement obsolètes, nous désirons conserver une solution utilisant ce microcontrôleur faible consommation mais compatible avec les APNs les plus récents. La solution consiste alors, en l'absence de maître USB, de simuler l'appui de touches. Il faut pour cela démonter l'APN afin de souder des fils sur les divers interrupteurs des touches : *cette opération annulera bien entendu toute garantie sur l'appareil démonté.*

Le composant que nous utiliserons pour simuler l'appui d'une touche – qui se traduit par la fermeture d'un contact électrique – est le relais analogique CMOS 4066. Ce composant, alimenté par une tension entre 3 et 15 V, fournit des paires de pattes dont la connexion entre elles est commandée numériquement. Un 4066 fournit quatre switchs et permet donc de simuler l'appui de 4 touches, par exemple les boutons de mise en marche et de prise de vue.

Nous avons utilisé pour nos expérience les APN les plus simples (sans partie mobile afin de mieux résister aux environnements climatiques extrêmes) avec une résolution intéressante (5 Mpixels) et les moins chers disponibles sur le marché : l'IT Works DSC551 est disponible pour 89 euros chez Darty (Fig. 4). Le concept a par ailleurs été validé sur un appareil réflexe haut de gamme Canon EOS 20D : dans ce cas, le 4066 court-circuit deux fils issus du déclencheur souple vendu séparéments (connecter au 4066 la tresse et la lame externe du déclencheur souple). Le principe de cette expérience peut de façon générale être adapté à tout instrument contrôlé par un interrupteur alimenté sous une tension faible (<15 V) afin d'automatiser la mise en marche d'instruments prévus pour un opérateur humain (tel que nous l'avons validé sur une micropipette motorisée dans le cadre de l'automatisation d'une expérience dangereuse pour un manipulateur humain).

[2] `www.arcturusnetworks.com/coldfire5272.shtml`

```
        .area code (ABS)
        .org 0h0000

RSTirq: ljmp    MAIN ; 3 bytes
IE0irq: nop nop nop nop nop nop nop nop  ; commence en 03
TF0irq: nop nop nop nop nop nop nop nop  ; commence en 0B
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop  ; RI+TI interrupt
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: nop nop nop nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop nop

MAIN:
    MOV     0hd7,#0h00  ; 16.77 MHz core clock for 19200
                ; baud rate but higher current consumption

    MOV     RCAP2H,#0hFF    ; config UART for 19200 baud (16.77 MHz core)
    MOV     RCAP2L,#-27     ; 19200
    MOV     TH2,#0hFF
    MOV     TL2,#-27
    MOV     SCON,#0h52
    MOV     T2CON,#0h34


;------------------ Camedia Init ---------------------------------
init_cam:
  mov   A,#0
  lcall SENDCHAR    ; init comm
  lcall GETCHAR     ; answers 0h15
  mov   dptr,#cmd17
  mov   r0,#12
  lcall snd_str     ; baud rate=19200
  lcall GETCHAR     ; ans: 0h06 (ACK)
  ret

noflash:
    mov dptr,#cmd_noflash
    mov r0,#12
    lcall   snd_str
    lcall   GETCHAR
    ret
```

```
camedia:
    mov dptr,#cmd02
    mov r0,#9
    lcall   snd_str     ; take pict
    ;lcall  GETCHAR     ; answer 0h06 (ACK)
    ;lcall  GETCHAR     ; ans: 0h05 (done)
    ret
;------------------ END OF Camedia Init ---------------------------------

;-----------------------------------------------------------------
SENDCHAR:       ; sends ASCII value contained in A to UART
        JNB     TI,SENDCHAR     ; wait til present char gone
        CLR     TI              ; must clear TI
        MOV     SBUF,A
        RET
;-----------------------------------------------------------------
GETCHAR:        ; waits for a single ASCII character to be received
                ; by the UART.  places this character into A.
iciget: JNB     RI,iciget
        MOV     A,SBUF
        CLR     RI
        RET
;-----------------------------------------------------------------
; dptr contains the starting point of the string to be sent
; R0 contains the length of the string, will be zero upon end of func.
snd_str:
    clr A
    movc    A,@A+dptr
    lcall   SENDCHAR
    inc     dptr
    djnz    r0,snd_str
    ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Minimal protocol for taking pictures with the Olympus Camedia C860L
; Initial RS232 communication baud rate MUST be 19200 (init + set speed)
                                ;/ cmd17=setspeed, arg=2=19200 bauds
cmd17: .db 0h1B,0h53,0h06,0h00,0h00,0h11,0h02,0h00,0h00,0h00,0h13,0h00
                                ;/ cmd02=snaphshot
cmd02: .db 0h1b,0h43,0h03,0h00,0h02,0h02,0h00,0h04,0h00
cmd_noflash:
       .db 0h1b,0h43,0h06,0h00,0h00,0h07,0h02,0h00,0h00,0h00,0h09,0h00
; command is always 1B 43 (except 53 for first packet in session)
; followed by 2 bytes=length and ending with checksum (2 bytes)
; http://photopc.sourceforge.net/protocol.html
```

Tab. 5 – Programme de contrôle d'un appareil photo numérique muni d'un port RS232 depuis un microcontrôleur de type ADuC814 compatible 8051. Les séries d'octets utilisées pour l'initialisation et le déclenhement d'une prise de vue sont visibles en fin de code (labels `cmd17`, `cmd02` et `cmd_noflash`). La transaction entre le microcontrôleur et l'APN est bidirectionnelle : l'appareil acquitte toute commande émise par le microcontrôleur.

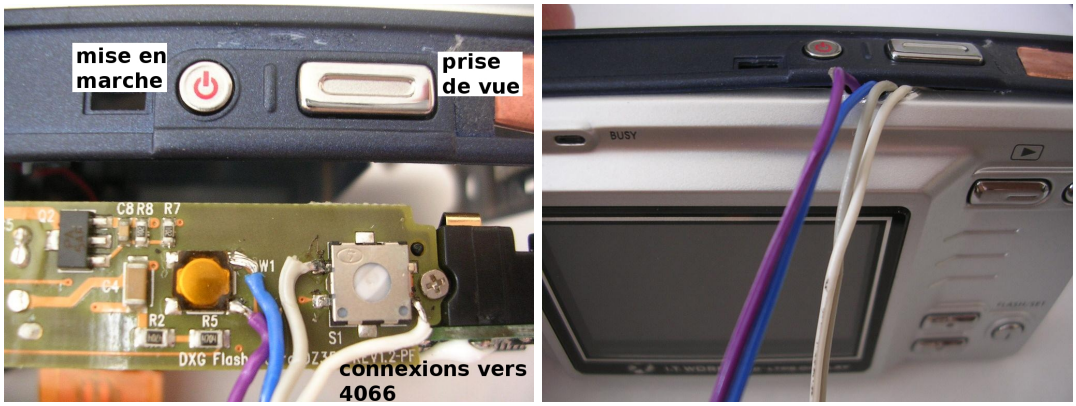

Fig. 4 – Connection sur les interrupteurs de mise en marche (gauche) et de prise de vue (droite) d'un APN – ici un IT Works DSC551 – en vue d'une commande numérique par un 4066. Éviter de toucher le condensateur du flash lors de ces opérations de soudure afin de ne pas subir une décharge électrique désagréable.

Nous verrons plus bas (label `fin` dans le tableau 9) que la simulation d'appui d'une touche se résume alors au passage d'un signal numérique issu du microcontrôleur, et connecté à une broche

de commande du 4066, du niveau bas au nivau haut (afin de court-circuiter les deux broches associées à cette commande), attendre un délai compatible avec celui qu'attendrait un opérateur (fonction `delai` dans le tableau 8), et finalement ouvrir l'interrupteur du 4066 en repassant le signal de commande au niveau bas.

6 Le contrôle du temps

Dans toutes les solutions citées précédemment, l'unité de calcul est continuellement en fonctionnement et consomme inutilement de l'énergie en attendant d'atteindre la condition de prise de vue. Une façon plus efficace de gérer l'énergie consiste à ne réveiller l'unité de calcul que lorsque le moment de prise de vue est venu, sous réserve que la séquence d'initilisation (*boot*) ne soit pas plus gourmande en énergie que l'attente : cette condition sera toujours vérifiée pour des captures d'images à intervalles de temps réguliers espacés dans une journée.

L'horloge temps-réel Maxim DS1305 [3] fournit un calendrier et une horloge comme tous les autres composants de ce type, mais propose en plus deux alarmes associées à deux signaux d'interruption (actifs bas) permettant de réveiller un périphérique lorsqu'une condition d'égalité entre un registre d'alarme et l'heure courante est vérifiée. De plus, une horloge temps réel permet d'aisément générer des signaux à des intervalles de temps longs (qui peuvent se compter en heures ou en jours) difficiles à générer avec les compteurs internes d'un microprocesseur [4], et ce avec une précision égale à celle du diapason à quartz sur lequel l'oscillateur de l'horloge est asservie.

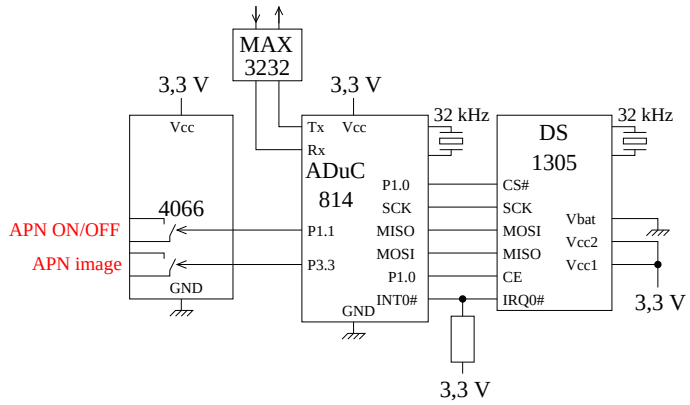

FIG. 5 – Schéma de principe du circuit centré sur le microcontrôleur ADuC814 chargé de programmer la RTC par son port SPI et de commander l'allumage et le déclenchement de prise de vue de l'APN par simulation de l'appui des touches appropriées. Le microcontrôleur est la plupart du temps en mode veille pour être réveillé aux moments appropriés par l'horloge temps-réel qui déclenche une interruption matérielle lorsqu'un de ses registres d'alarme est égal à l'heure courante.

Notre stratégie sera donc la suivante (Fig. 5) : à l'allumage, un microcontrôleur reçoit de l'utilisateur les paramètres de programmation de l'horloge temps réel (*Real Time Clock* : RTC) et initialise l'horloge avec ces données. La mise en mode sommeil du microcontrôleur réduit alors considérablement sa consommation (dans son mode de sommeil le plus profond l'oscillateur est arrêté et la consommation associée aux transitions d'état dans le processeur est alors à son minimum, avec seulement la rétention des informations dans les registres et la RAM). La génération d'une impulsion en sortie d'alarme de la RTC déclenche une interruption matérielle du processeur qui se réveille alors pour exécuter la fonction suivant l'instruction de mise en veille – fonction de

[3] références 795987 et 9726209 pour respectivement 4,87 et 8,18 euros chez Farnell

[4] voir à ce sujet la note d'application SLAA076A de Texas Instruments, `focus.ti.com/lit/an/slaa076a/slaa076a.pd`

prise de vue et de réinitialisation de la RTC avec la programmation de l'alarme pour le réveil suivant – avant de se rendormir.

Afin d'atteindre des autonomies de l'ordre de l'année sur une pile, la consommation du circuit doit être réduite à son strict minimum en sélectionnant une RTC de faible consommation (typiquement inférieure au mA), prendre soin de réduire le rapport cyclique réveil/sommeil du processeur et de réduire sa consommation lors des phases d'inactivité, éviter les diodes et autres résistances de tirage de valeur trop faible qui induisent des courants de fuite inutiles, et finalement désactiver les composants inutiles. Dans cette dernière classe d'optimisation, nous avons par exemple découvert qu'un convertisseur de niveaux logiques MAX3232 consomme 5 mA même inactif : nous avons donc pris soin d'ajouter un jumper sur son alimentation pour le déconnecter après la phase de programmation du microcontrôleur et d'initialisation de la RTC. Une solution plus élégante est l'utilisation du MAX3222 qui possède une broche de désactivation.

Nous proposons dans cet exemple d'illustrer l'utilisation du microcontrôleur ADuC814 pour simuler l'appui des touches de mise en marche et de prise de vue d'un APN, sa mise en mode faible consommation et son réveil sur le déclenchement d'une interruption externe. Ce microcontrôleur est de plus équipé de ports RS232 et SPI pour la réception des consignes de prises de vues fournies par l'utilisateur et la communication avec la RTC respectivement.

Une utilisation efficace pour des prises de vues synchrones par plusieurs montages commandés par des RTCs distinctes nécessite de sélectionner un algorithme de synchronisation. Nous proposons d'initialiser toutes les RTC avec l'heure de l'ordinateur servant à programmer les paramètres dans les circuits situés en des points distincts géographiquement (l'horloge de plusieurs ordinateurs pouvant être synchronisée par `ntp` [5] ou GPS). Chaque jour, les RTCs ont pour consigne de se réveiller à une heure prédéterminée pour amorcer la séquence de prises de vues qui se poursuit à intervalles de temps réguliers dans la journée pour s'achever après un nombre prédéterminé d'images prises. Ainsi nous garantissons que chaque jour tous les appareils se réveillent à la même heure, et ne prennent que des photos lors de conditions d'illumination pertinentes en fonction des réglages des instruments de prise de vue (uniquement de jour ou uniquement de nuit par exemple).

Lors de la mise sous tension, le microcontrôleur s'initialise par la routine visible dans le tableau 6. Nous allons utiliser l'interruption matérielles INT0# afin de réveiller le microcontrôleur de son sommeil (faible consommation électrique) dans lequel il se place entre deux prises de vues : l'emplacement à l'adresse de gestion de cette interruption (ISR en 0x03) contient donc simplement un retour d'interruption (`RTI`) qui, au déclenchement de l'interruption, permettra au programme de continuer séquentiellement son exécution à l'instruction qui suit la mise en veille du microcontrôleur. Nous prenons donc soin d'activer cette interruption (bit EX0 du registre Interrupt enable `EX0=1`) ainsi que les interruptions en général (Global Interrupt Enable `EA=1`). De plus, le port RS232 est initialisé à 9600 bauds puisque nous l'utiliserons pour recevoir les paramètres d'initialisation de la RTC.

Une fois ces tâches d'initialisation réalisées, nous passons à la phase de configuration de la RTC. Un PC se charge de prendre l'heure se sa propre horloge pour la transférer au circuit : l'heure du PC est obtenue par `time_t t ;time(& t) ;` puis est convertie en un format acceptable par la RTC de la forme année, mois et jour par la fonction `tm=localtime(&t) ;` qui renvoie un `struct tm *tm ;`. Cette structure de données contient alors les informations que nous transférons au microcontrôleur, à savoir `tm->tm_min`, `tm->tm_hour`, `tm->tm_mday`, `tm->tm_mon+1`, `tm->tm_year-100`. Les fonctions de communication par le port SPI (tableau 8) vont nous permettre de transférer ces infomations de l'ADuC814 à la RTC, non sans avoir au prálable pris soin de faire passer le signal d'activation de la RTC nommé CE (*attention* : ce signal a la particularité d'être actif au niveau haut) de bas à haut afin d'annoncer le début d'une transaction. Cette opération est développée dans la fonction `fillram` du code présenté dans le tableau 7, code dans lequel nous avons mis sous forme de commentaire les fonctions d'initialisation des mois, jour, ... minutes afin d'alléger la présentation. Toutes ces fonctions sont en fait identiques à la fonction de définition de l'année courante présentée dans son intégralité, en remplaçant simplement le registre de destination dans la RTC par le registre explicité dans le commentaire.

[5] http ://www.ntp.org/

```
        PCON    = 0h87
        battery = P1.1 ; mise en marche de la camera
onoff = P3.3 ; declenchement de l'APN
        RTCcs   = P1.0

        .area code (ABS)
        .org 0h0000

RSTirq: ljmp    MAIN ; 3 bytes
IE0irq: reti nop nop nop nop nop nop nop    ; commence en 03
TF0irq: nop nop nop nop nop nop nop nop     ; commence en 0B
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop     ; RI+TI interrupt
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: nop nop nop nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop
```

```
MAIN:
;    MOV     0hd7,#0h00  ; 16.77 MHz core clock for 19200
                ; baud rate but higher current consumption

     MOV     RCAP2H,#0hFF    ; config UART for 9600 baud (2 MHz core)
     MOV     RCAP2L,#-7      ; 9600
     MOV     TH2,#0hFF
     MOV     TL2,#-7
     MOV     SCON,#0h52
     MOV     T2CON,#0h34

     SETB    0hAF            ; enable interrupts -- EA=0hAF
     SETB    0hA8            ; enable EX0 interrupts - EX0=0hA8

     mov     0h9c,#0h01  ; cfg814=0h9C
     mov     0hF8,#0h3D  ; SPICON=0hF8

     clr     RTCcs
     clr     battery     ; open relay
     clr     onoff    ; open relay
```

Tab. 6 – Initialisation de l'ADuC814 avec la définition d'un vecteur d'interruption associé à l'interruption matérielle servant à réveiller le microcontrôleur par un signal issu de la RTC.

Nous passons donc comme paramètres à la RTC le triplet d'octets représentant la date, le triplet correspondant à l'heure actuelle et le triplet correspondant à l'horaire de réveil. Suivent l'intervalle de temps entre deux prises (heure et minute) puis le nombre de photos à prendre chaque jour. Ces six derniers octets sont stockés dans un petit espace mémoire (RAM) fourni par la RTC toujours maintenu sous tension. Ces valeurs de référence seront conservées inchangées pour consultation à chaque déclenchement de l'alarme. En effet, rappelons que nous avons choisi de débuter les séquences de prises de vues à une certaine heure prédéterminée chaque jour, puis d'incrémenter d'un intervalle d'heures/minutes un nombre prédéterminé de fois au cours de cette journée. À la fin de la journée, les registres d'alarme et le compteur de prises de vues déjà effectuées doivent être réinitialisés en prévision de la nouvelle journée à venir : c'est le rôle de la fonction `fillinitval` du code présenté dans le tableau 7. Cette fonction cherche les valeurs dans la RAM de la RTC et les transfère dans le registre d'alarme0 de la RTC, et intialise le compteur d'images déjà prises (emplacement mémoire 0xA5), compteur qui est décrémenté à chaque prise de vue jusqu'à atteindre 0 indiquant la fin de la journée.

Finalement, le dernier point qui nous manque est la définition de la nouvelle alarme lorsque le microcontrôleur se réveille mais que la fin de journée n'est pas encore atteinte. Dans ce cas, il nous faut lire l'horaire courant (heure/minutes), les incrémenter des intervalles de temps définis par l'utilisateur, prendre soin de compenser les dépassements de capacité (heure>24 et surtout minutes>60) et stocker les résultats dans les registres d'alarme0. Ces fonctions sont implémentées dans le code du tableau 9. La seule subtilité de ce code est la conversion de l'information stockée au format BCD dans la RTC vers un format binaire que nous avons choisi d'utiliser pour la gestion de nos opérations arithmétiques, le résultat étant converti de nouveau en format BCD après avoir effectué les additions et compensé les dépassements (fonctions `BCD2BIN` et `BIN2BCD` dans le code du tableau 8).

L'ADuC814 ne présente que peu d'options pour sélectionner un mode de faible consommation d'énergie : étant donné qu'aucune activité n'est requise du processeur entre deux prises de vues, nous avons activé le mode de consommation dit *power down*. Dans ce mode, toutes les horloges du microcontrôleur sont arrêtées, les sorties numériques et registres internes conservent leurs positions, et nous activons le réveil par déclenchement de l'interruption matérielle INT0#. Ces résultats s'obtiennent en définissant `PCON=0x23` après la prise de vue. Un microcontrôleur dédié aux applications fonctionnant sur batteries tel que le Texas Insutruments MSP430 offre beaucoup plus de précision dans la désactivation des périphériques pour finalement atteindre une consommation encore plus faible.

Le résultat de ce travail est un instrument relativement compact – à peine plus grand que l'appareil photo numérique qui sert aux prises de vues – capable de fonctionner en totale autonomie pendant près d'un an sur une pile de capacité 11 A.h. L'APN n'étant allumé que quelques secondes chaque jour, sa consommation moyenne est négligeable devant celle du circuit de contrôle

```
;------------------ RTC access routines ----------------------------
; placer 0x80 dans les champs qui doivent etre executes a chaque fois

    MOV P1,#0hFE
    LCALL   fillram ; recoit les params de config de l'utilisateur
            ; 5 chars = heure:min ini, heure:min inc, nb/jour
    setb    RTCcs
    MOV     A,#0h8F ; fill control register : 8F <- 0x05
    LCALL   SENDSPI
    MOV     A,#0h05 ;
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h8A ; fill day: take pict every day
    LCALL   SENDSPI
    MOV     A,#0h80 ;
    LCALL   SENDSPI
    clr     RTCcs

    lcall   fillinitval
    setb    RTCcs
    MOV     A,#0h07     ; read alarm0 register to restart
    LCALL   SENDSPI
    LCALL   READSPI

    clr     RTCcs
    clr     battery     ; open relay
    clr     onoff    ; open relay

; SUITE DU PROGRAMME PRINCIPAL ICI : APRES CA ROUTINES DE RTC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; get from RS232 port the current time/date, alarm start and time interval -> RAM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
fillram:    ; we store restart time every day (hour:min) : A0 A1
            ; time interval between pictures (hour:min)  : A2 A3
            ; number of picture taken per day (one char) : A4
    ;;;;;; current year ;;;;;;;;;;;;;;;;;
    setb    RTCcs
    MOV     A,#0h86 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   BIN2BCD
    LCALL   SENDSPI ; current year
    clr     RTCcs

;;; IDEM pour registre #0h85 lu sur RS232 = current month
;;; IDEM pour registre #0h84 lu sur RS232 = current day
;;; placer 1 dans registre #0h82 = current day of week
;;; IDEM pour registre #0h82 lu sur RS232 = current hours
;;; IDEM pour registre #0h81 lu sur RS232 = current minutes

    setb    RTCcs ;;;;;;;;;;;;;;; hours ;;;;;;;;;;;;;;;;;
    MOV     A,#0hA0 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI ; init time hour
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; minutes ;;;;;;;;;;;;;;;;
    MOV     A,#0hA1 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; interval ;;;;;;;;;;;;;;;
    MOV     A,#0hA2 ; inc hour
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; interval ;;;;;;;;;;;;;;;
    MOV     A,#0hA3 ; inc min
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; number/day ;;;;;;;;;;;;
    MOV     A,#0hA4 ; pict/day
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs
    RET         ; END of fillram

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; set alarm 0 to the initial time and the counter to its initial value
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
fillinitval:
    setb    RTCcs
    MOV     A,#0h20 ; read init hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    LCALL   BIN2BCD
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h89 ; fill hours
    LCALL   SENDSPI
    mov     A,R0
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCCs
    MOV     A,#0h21 ; read init minutes
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    LCALL   BIN2BCD
    push    acc

    setb    RTCcs
    MOV     A,#0h88 ; fill minutes
    LCALL   SENDSPI
    pop acc
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h87 ; fill seconds
    LCALL   SENDSPI
    mov     A,#00
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h24 ; read init number
    LCALL   SENDSPI
    LCALL   READSPI
    push    acc
    clr     RTCcs
    setb    RTCcs
    MOV     A,#0hA5 ; fill number
    LCALL   SENDSPI
    pop acc
    LCALL   SENDSPI
    clr     RTCcs
    ret     ; fillinitval
```

Tab. 7 – Communication par port SPI entre l'ADuC814 et la RTC DS1305 lors de son initialisation : transfert de la date et heure actuelle et des consignes de réveil.

(ADuC814 en mode veille et RTC). Une amélioration possible de ce circuit est de découpler au maximum les sources d'énergie afin de palier au mieux aux défaillances potentielles d'une pile : l'appareil de prises de vues peut être alimenté totalement indépendamment des autres circuits, et nous n'avons pas pris avantage ici de la possibilité d'utiliser une batterie de sauvegarde sur la RTC afin de ne pas perdre les données en cas de coupure de l'alimentation principale. En effet, en cas de dysfonctionnement de la batterie alimentant le microcontrôleur, celui-ci se met directement en attente des commandes sur le port RS232 et interrompt donc la séquence de prises de vues. Il nous faudrait donc encore modifier le contenu de la RAM de la RTC afin d'y mettre un indicateur pour que l'ADuC814 poursuive son activité de déclenchement de l'APN si la RTC a déjà été initialisée. Enfin, découpler les alimentations permettrait éventuellement de placer des accumulateurs sur les alimentations qui ne sont pas stratégiques (l'APN par exemple) afin de les

```
;-------------------------------------------------------------------
SENDCHAR:       ; sends ASCII value contained in A to UART
        JNB     TI,SENDCHAR     ; wait til present char gone
        CLR     TI              ; must clear TI
        MOV     SBUF,A
        RET
;-------------------------------------------------------------------
SENDSPI:    ; sends the content of A to the SPI port
    MOV     0hF7,A          ; trigger data transfer: SPIDAT=0hF7
icispi: JNB 0hFF,icispi
    CLR 0hFF        ; clear ISPI
    RET
;-------------------------------------------------------------------
READSPI:    ; sends the content of A to the SPI port
    MOV A,#0hff          ; generate clocks for reception
    LCALL   SENDSPI

    mov     A,0hF7      ; MOV A,SPIDAT: read resulting value
    CJNE    A,#0hff,readend ; we only continue if the MMC answers '0x01'
    SJMP    readspi
readend:RET
;-------------------------------------------------------------------
GETCHAR:        ; waits for a single ASCII character to be received
                ; by the UART.  places this character into A.
iciget: JNB     RI,iciget
        MOV     A,SBUF
        CLR     RI
        RET
;-------------------------------------------------------------------
BIN2BCD:
        push    b                       ; Save B
        mov     b,#10                   ; Divide By 10
        div     ab                      ; Do Divide
        swap    a                       ; Move Result To High Of A
        orl     a,b                     ; OR In Remainder
        pop     b                       ; Recover B
        ret                             ; Return To Caller
;-------------------------------------------------------------------
BCD2BIN:
        push    b                       ; Save B
        push    acc                     ; Save Value
        anl     a,#0hf0                 ; Keep High Bits
        swap    a                       ; Get To Low
        mov     b,#10                   ; 10 Decimal
        mul     ab                      ; Multiply
        mov     b,a                     ; Store In B
        pop     acc                     ; Recover BCD Value
        anl     a,#0h0f                 ; Keep Low Nybble
        add     a,b                     ; Add In High
        pop     b                       ; Recover B
        ret                             ; Return To Caller
;-------------------------------------------------------------------
delai: ; si R1=#60 alors on attend 17 secondes
DLY2:   MOV     R2,#0hff        ; Set up delay loop0
DLY1:   MOV     R3,#0hff         ; Set up delay loop1
dly:    DJNZ    R3,dly           ; Dec R3 & Jump here until R3 is 0
        DJNZ    R2,DLY1          ; Dec R2 & Jump DLY1 until R2 is 0
        DJNZ    R1,DLY2
        RET                      ; Return from subroutine
```

Tab. 8 – Quelque routines de communication par protocoles RS232 et SPI nécessaires pour l'échange d'informations avec l'utilisateur et la RTC respectivement.

recharger par panneaux solaires, sans mettre en péril le fonctionnement de l'ensemble du montage en cas de décharge des accumulateurs puisque l'électronique de contrôle continue à fonctionner sur piles. Diverses stratégies de redondance des alimentations sont alors envisageables pour rendre ce montage totalement autonome pour une durée idéalement illimitée, utilisant notamment l'option de recharge de batterie du DS1305.

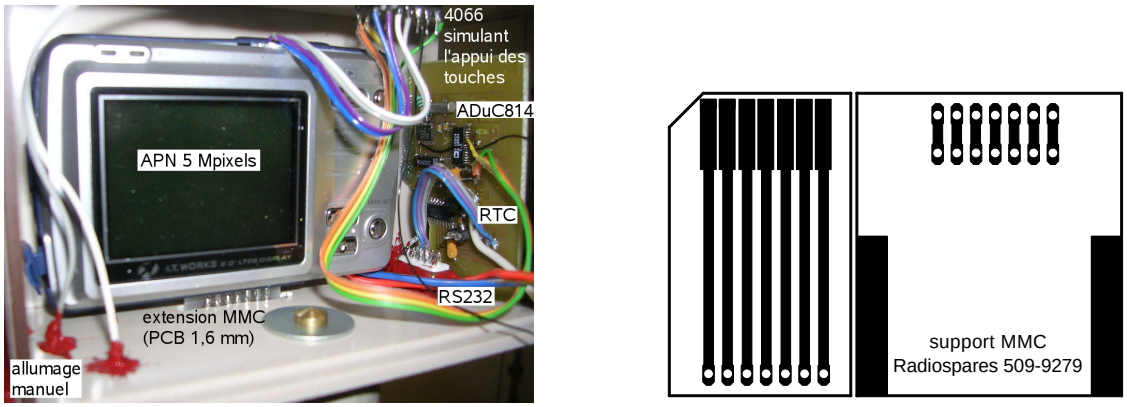

Fig. 6 – L'APN modifié monté dans son boîtier étanche, avec le circuit électronique de contrôle visible sur la droite. Il ne manque que la pile pour fermer la boite. Noter l'extension de la carte mémoire MMC réalisée en circuit imprimé de 1,6 mm d'épaisseur selon le schéma joint (la largeur du circuit imprimé remplaçant la MMC doit être de 24 mm), qui permet de placer la carte mémoire à l'extérieur du boîtier contenant l'APN et donc de remplacer la carte de stockage des images sans toucher à l'orientation de l'appareil photo. Les deux fils à gauche du boîtier débouchent vers l'extérieur et sont montés en parallèle du 4066 : leur court circuit avec un objet conducteur permet la mise en marche ou l'extinction de l'APN lors de l'installation pour régler l'orientation en visant grâce à l'image visible sur l'écran LCD.

```
fin:
    mov     PCON,#0h23  ; sleep, enable wakeup triggered by INT0
    clr     onoff    ; desactive prise de vue

    LCALL   reinit ; on reinitialise tout de suite le RTC

    setb    battery     ; allume l'APN
    mov     r1,#2
    LCALL   delai       ; duree d'appui sur l'interrupteur de mise en route
    clr     battery
    mov     r1,#5
    LCALL   delai       ; attente pour l'ajustement de la luminosite
;    LCALL   dumpall

    setb    onoff     ; appui sur la prise de vue
    mov     r1,#2      ; #60=17 seconde
    LCALL   delai       ; duree d'appui sur la prise de vue
    clr     onoff
    mov     r1,#2      ; attente de sauvegarde de la prise de vue
    LCALL   delai       ;

    setb    battery     ; appui sur l'interrupteur de mise en marche APN
    mov     r1,#3
    LCALL   delai       ;
    clr     battery     ; on a fini d'etteindre l'APN
    SJMP    fin

; lire heure/min actu et ajouter delta heure/min
; compenser pour des min > 60 et des heures > 24
; decrementer compteur et si compteur == 0 alors mettre heure a init val
reinit:setb RTCcs
    MOV     A,#0h07 ; read alarm0 register to restart
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h25 ; number of pictures taken so far
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    dec     a    ; DECREMENTER ET RESTOCKER
    jnz     continue
    lcall   fillinitval
    ret
continue:       ; non nul donc on n'est pas en fin de journee
    push    acc
    setb    RTCcs
    MOV     A,#0hA5 ; number of pictures still to be taken
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI ; store
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h22 ; read inc hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h02 ; read current hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    LCALL   BCD2BIN

    ADD     A,R0
    MOV     R0,A
    SUBB    A,#24   ; C is set if A<24
    JNC     finh
    MOV     A,R0    ; restore old value of A since result is <0
finh:LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h89 ; fill alarm hours
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h23 ; read inc min
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h01 ; read current min
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    LCALL   BCD2BIN

    ADD     A,R0
    MOV     R0,A
    SUBB    A,#60   ; C is set if A<60
    JNC     corrigeh
    MOV     A,R0    ; restore old value of A since result is <0
finm:
    LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h88 ; fill alarm min
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs
    RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
corrigeh:
    push    acc
    setb    RTCcs
    MOV     A,#0h09 ; read alarm hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    LCALL   BCD2BIN
    INC A
    mov     R0,A
    SUBB    A,#24   ; C is set if A<24
    JNC     finhc
    MOV     A,R0    ; restore old value of A since result is <0
finhc:
    LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h89 ; fill alarm hours
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs
    pop     acc
    sjmp    finm
```

Tab. 9 – Routine définissant la nouvelle alarme après une prise de vue : s'il ne s'agit pas de la dernière photo de la journée, l'horaire courant est incrémenté des intervalles en heures et en minutes pré-définis pour déterminer les nouvelles valeurs des registres d'alarme, sinon le décompte des prises de vues est réinitialisé et l'horaire de première prise de vue le lendemain est placé dans les registres d'alarme.

7 Conclusion

Nous avons abordé diverses méthodes de déclenchement automatique d'instruments permettant la capture d'images, avec des contraintes de consommation électrique décroissantes mais de modification à l'appareil photo numérique croissante. Nous sommes partis d'un PC équipé d'une carte d'acquisition vidéo pour une capture périodique d'images ou lorsqu'un événement se produit dans le champ de vision, pour ensuite déclencher un APN par les ports prévus à cet effet (USB ou RS232) pour finalement simuler l'appui de touches par des interrupteurs analogiques commandés par microcontrôleur. Cette dernière solution ne consomme que quelques centaines de microwatts et offre le plus de perspectives pour une utilisation à long terme d'un circuit autonome installé en

milieu hostile où la présence de l'opérateur n'est qu'épisodique.

Remerciements

La carte FOX a été gracieusement prêtée par le laboratoire FEMTO-ST/LPMO de Besançon. L'ensemble des composants Analog Devices (microcontrôleurs ADuC814 et convertisseurs de niveaux RS232 ADM3202 et ADM3222) ont été fournis gratuitement dans le cadre du programme d'offres d'échantillons de Analog Devices (`www.analog.com`).

Références

[1] Projet Aurore, *Films accélérés : méthodes d'acquisition et traitements*, Bulletin de l'Union de Physiciens n.842 (Mars 2002), p.543, disponible à `http://jmfriedt.free.fr/bup_films.pdf`, et un exemple d'animation obtenu en 1997 par Connectix Quickcam noir&blanc qui y est citée à `http://friedtj.free.fr/jupitereclipse.mpg`

[2] Pilotes pwc pour webcam USB : `http://www.saillard.org/linux/pwc/`

[3] `transcode` est disponible à `http://www.transcoding.org/cgi-bin/transcode`

[4] `mencoder` est disponible à `http://www.mplayerhq.hu/design7/news.html`

[5] Une façon triviale de passer d'une source de courant à une autre tout en chargeant une batterie, le tout uniquement avec des composants passifs : `http://www.cq-vhf.com/Switching.pdf`

[6] J.-M Friedt, S. Guinot & E. Carry, *Introduction au Coldfire 5272*, GNU/Linux Magazine France **73** (Juin 2005), pp.26-33